Licenciatura em Computação

APOSTILA DE

LÓGICA DE PROGRAMAÇÃO

Prof. Walteno Martins Parreira Júnior

www.waltenomartins.com.br

waltenomartins@iftm.edu.br

2022

GPETEC - Grupo de Pesquisa em Educação, Tecnologia e Ciências – IFTM UdiCentro

Linha de pesquisa: Desenvolvimento de aplicativos tecnológicos e softwares educacionais

Autor

Walteno Martins Parreira Júnior – Doutorando em Educação (UFTM), Mestre em Educação (UFU), Professor da Licenciatura em Computação e do Técnico Integrado em Programação de Jogos Digitais do IFTM Campus UdiCentro, Membro do GPETEC e Coordenador de Projetos de Pesquisa e Extensão em Informática Aplicada à Educação.

Currículo Lattes: http://lattes.cnpq.br/4647904741241414

2022

SUMÁRIO

1 - INTRODUÇÃO À LÓGICA

1.1 - Origens

A lógica iniciou seu desenvolvimento na Grécia Aristóteles (384 – 322 AC) e os antigos filósofos gregos passaram a usar em suas discussões sentenças enunciadas nas formas afirmativa e negativa, resultando em grande simplificação e clareza.

Em 1847, Augustus DeMorgam (1806-1871) publicou o tratado Formal Logic. Em 1848, George Boole (1815-1864) escreveu The Mathematical Analysis of Logic e depois publicou um livro sobre o que foi denominado posteriormente de Álgebra de Boole.

Em 1879, Gotlob Frege (1848-1925) contribuiu no desenvolvimento da lógica com a obra Begriffsschrift. As ideias de Frege só foram reconhecidas pelos lógicos mais ou menos a partir de 1905. A escola italiana, que desenvolveu quase toda simbologia da matemática utilizada atualmente, é composta de Giuseppe Peano (1858-1932) e também por Burali-Forti, Vacca, Pieri, Pádoa, Vailati, etc.

Bertrand Russell (1872-1970) e Alfred North Whitehead (1861-1947) iniciam o atual período da lógica com a publicação da obra Principia Mathematica no início do século XX.

Também contribuem para o estágio atual, David Hilbert (1862-1943) e sua escola alemã com von Neuman, Bernays, Ackerman e outros.

Em 1938, Claude Shannon mostrou a aplicação da álgebra de Boole na análise de circuitos de relês.

Podemos dizer que a lógica estuda as condições objetivas e ideais para justificar a verdade e não cuida da própria verdade. Ela estuda as condições formais para justificar a verdade, isto é, as condições que o pensamento deve preencher para ser coerente consigo mesmo e demonstrar a verdade já conhecida. A lógica estuda as relações do pensamento consigo mesmo para possibilitar a construção de um contexto correto de justificação, isto é, para a definição argumentativa de premissas corretamente dispostas para uma conclusão justificada.

Uma das condições para se ter a verdade demonstrada, e portanto justificada, é que o pensamento se ja coerente consigo mesmo, isto é, que siga as leis da razão expressa linguisticamente.

1.2 – Definições

Lógica é uma palavra que define a ciência do raciocínio. Outro conceito de lógica é “o estudo dos métodos e princípios usados para distinguir o raciocínio correto do incorreto”. Essa ciência abrange vários conceitos, dentre eles a argumentação, a matemática e a informática.

A ***lógica***, como estudo e identificação das formas válidas e corretas na linguagem, também se dedica a identificar e qualificar aquilo que não possui uma validade formal coesa e correta. A palavra que nomeia essas situações de não correção da forma daquilo que foi enunciado pela linguagem é falácia. As falácias são, grosso modo, proposições sem sentido, sem encadeamento lógico entre os fatos enunciados ou sem nexos causais que expliquem de maneira completa e correta os efeitos que constam nos enunciados das frases analisadas (PORFÍRIO, 2019).

A ***lógica de programação*** consiste na elaboração de sequências lógicas. Também pode ser definida como o modo como se escreve um programa de computador, um algoritmo.

Podemos dizer que ***algoritmo*** é o conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas (MOREIRA, 2017).

Ele não responde a pergunta “o que fazer?”, mas sim “como fazer”. Em termos mais técnicos, um algoritmo é uma sequência lógica, finita e definida de instruções que devem ser seguidas para resolver um problema ou executar uma tarefa (PEREIRA, 2009).

Um ***algoritmo*** é uma sequência de passos para se executar uma função. Um exemplo de algoritmo, fora da computação, é uma receita de bolo.

Na receita de bolo, devem-se seguir os passos para o bolo ficar pronto e sem nenhum problema. Na informática, os programadores escrevem as “receitas de bolo” (algoritmos) de modo que o computador leia e entenda o que deve ser feito, ao executar o algoritmo. Para isto é necessário uma linguagem de programação (PACIEVITCH).

Os ***algoritmos*** mostram ao computador o que ele deve fazer em cada sequência lógica. Os algoritmos são escritos usando a linguagem de programação que pode ser de alto ou de baixo nível.

A ***linguagem da programação*** é um conjunto de palavras que tem significados específicos, razão pela qual é imprescindível definir exatamente o que se deseja, a fim de que a máquina saiba assimilar corretamente cada função e comando, apresentado pelo algoritmo (MOISES,).

2 - LÓGICA MATEMÁTICA

2.1 – Proposições simples (atômicas)

Definição: chama-se **proposição simples**, a proposição que não contém nenhuma outra proposição como parte integrante de si.

Exemplos:

P: Carlos é careca.

Q: Pedro é estudante.

R: O galo põe ovos.

2.2 – Conectivos lógicos

Definição: são palavras ou frases que são usadas para formar novas proposição a partir de outras proposições.

Os conectivos usuais em lógica matemática são:

Conectivo	Palavras	Símbolo
Negação	Não	**~ ou ¬**
Conjunção	E	$\wedge$
Disjunção	Ou	$\vee$
Condicional	Se ... então ...	$\rightarrow$
Bicondicional	... Se, e somente se ...	$\leftrightarrow$

2.3 – Proposições compostas

Definição: são as proposições formadas por duas ou mais proposições simples e ligadas pelos conectivos lógicos.

Exemplos:

P: Carlos é careca **e** Pedro é estudante.

Q: Carlos é careca **ou** Pedro é estudante.

R: **Se** Carlos é careca, **então** Carlos é infeliz.

S: Carlos é careca **se, e somente se** Pedro é estudante.

2.4 - Tabelas-verdade e conectivos.

Segundo o Princípio do terceiro excluído, toda proposição simples P é verdadeira ou é falsa, isto é, tem valor lógico V (verdade) ou o valor lógico F (falsidade). Em uma proposição composta, a determinação do seu valor lógico é feito segundo o princípio: “O valor lógico de qualquer proposição composta

depende unicamente dos valores lógicos das proposições simples componentes, ficando por eles univocamente determinado.”

Para aplicar este princípio na prática, recorre-se ao uso do dispositivo denominado **Tabela-verdade**, que apresenta todos os possíveis valores lógicos da proposição composta correspondentes a todas as possíveis atribuições de valores lógicos às proposições simples correspondentes.

O número de linhas da tabela-verdade de uma proposição composta está em função do número de proposições simples que a compõem. A tabela-verdade de uma proposição composta com **n** proposições simples contém $\mathbf{2^n}$ linhas.

Portanto, podemos observar:

a) Para uma proposição simples P, o numero de linhas da tabela-verdade será: $2^1 = 2$, representando na tabela-verdade:

P
V
F

b) Para uma proposição composta cujas proposições simples componentes são P e Q, o número de linhas da tabela-verdade será: $2^2 = 4$, que será representada:

P	**Q**
V	V
V	F
F	V
F	F

c) Para uma proposição composta cujas proposições simples componentes são P, Q e R, o número de linhas da tabela-verdade será: $2^3 = 8$, que será representada:

P	**Q**	**R**
V	V	V
V	V	F
V	F	V
V	F	F
F	V	V
F	V	F
F	F	V
F	F	F

Observe que os valores lógicos V e F se alternam de quatro (4) em quatro para a primeira proposição (P), de dois (2) em dois para a Segunda proposição (Q) e de um (1) em um para a terceira proposição (R).

2.4.1 – Operação Lógica Negação

Definição: chama-se negação de uma proposição P a proposição representada por "Não P", cujo valor lógico é a verdade (V) quando P é falsa e a falsidade (F) quando P é verdadeira.

Simbolicamente, a negação de P indica-se com a notação "~P", que se lê: "**NÃO P**"

A tabela-verdade:

P	**~P**
V	F
F	V

v(~P) = ~ v(P)

Para uma proposição P, podemos formar a sua negação de qualquer um dos seguintes modos:

- ✓ "não é verdade que P"
- ✓ "é falso que P"
- ✓ "não" em P

Exemplo:

P: Lídia é estudiosa.

~P: Não é verdade que Lídia é estudiosa.

~P: É falso que Lídia é estudiosa.

~P: Lídia não é estudiosa.

Q: João não foi ao cinema.

~Q: É falso que João não foi ao cinema.

Observações: Pode-se ter algumas variações, por necessidades da língua portuguesa, por exemplo:

P: Todos os homens são elegantes.

~P: Nem todos os homens são elegantes.

Q: Nenhum homem é elegante.

~Q: Algum homem é elegante.

2.4.2 – Operação Lógica Conjunção

Definição: chama-se conjunção de duas proposições P e Q a proposição representada por "P e Q", cujo valor lógico é a verdade (V) quando as proposições P e Q são ambas verdadeiras e a falsidade (F) nos demais casos.

Simbolicamente, a conjunção de duas proposições P e Q indica-se com a notação "P $\wedge$ Q ", que se lê "**P E Q**". A tabela-verdade:

P	Q	$P \wedge Q$
V	V	V
V	F	F
F	V	F
F	F	F

$v(P \wedge Q) = v(P) \wedge v(Q)$

Exemplo:

P: Pitágoras era Grego Q: Descartes era Francês

$P \wedge Q$: Pitágoras era Grego e Descartes era Francês.

$v(P) = V$ e $v(Q) = V \Rightarrow v(P \wedge Q) = V$

2.4.3 – Operação Lógica Disjunção

Definição: chama-se disjunção de duas proposições P e Q a proposição representada por "P ou Q", cujo valor lógico é a verdade (V) quando ao menos uma das proposições P e Q é verdadeira e a falsidade (F) quando as proposições P e Q são ambas falsas.

Simbolicamente, a conjunção de duas proposições P e Q indica-se com a notação "$P \vee Q$ ", que se lê "**P OU Q**".

A tabela-verdade:

P	Q	$P \vee Q$
V	V	V
V	F	V
F	V	V
F	F	F

$v(P \vee Q) = v(P) \vee v(Q)$

Exemplo:

P: A cidade de Paris é a capital da França

Q: O sol é um satélite artificial da terra

$P \vee Q$: A cidade de Paris é a capital da França ou o sol é um satélite artificial da terra.

$v(P) = V$ e $v(Q) = F \Rightarrow v(P \vee Q) = V$

Na linguagem coloquial a palavra ou tem dois sentidos. Exemplifiquemos:

P: Amilton é bombeiro ou eletricista.

Q: Rosa é mineira ou goiana.

Na proposição Q (Rosa é mineira ou goiana), está afirmando que somente uma das proposições é verdadeira, pois não é possível ocorrer as duas coisas: Rosa ser mineira e goiana ao mesmo tempo.

Na proposição P, diz-se que o ou é inclusivo, já na proposição Q diz-se que o ou é exclusivo. Logo a proposição P é uma disjunção inclusiva ou simplesmente disjunção; e a proposição Q é a disjunção exclusiva.

A disjunção exclusiva de duas proposições P e Q é a proposição composta "P $\veebar$ Q", que se lê: "ou P ou Q" ou se lê "P ou Q, mas não ambos". É a <u>falsidade</u> (F) quando o valor lógico das proposições P e Q forem ambos verdadeiros ou ambos falsos e a <u>verdade</u> (V) quando P e Q tem valores lógicos diferentes.
A tabela-verdade:

P	Q	P $\veebar$ Q
V	V	F
V	F	V
F	V	V
F	F	F

$v(P \veebar Q) = v(P) \veebar v(Q)$

2.4.4 - Tabela Verdade dos conectores

Resumindo, tem-se a seguinte tabela verdade:

p	q	p $\wedge$ q	p $\vee$ q	p $\veebar$ q	p $\rightarrow$ q	p $\leftrightarrow$ q
V	V	V	V	F	V	V
V	F	F	V	V	F	F
F	V	F	V	V	V	F
F	F	F	F	F	V	V

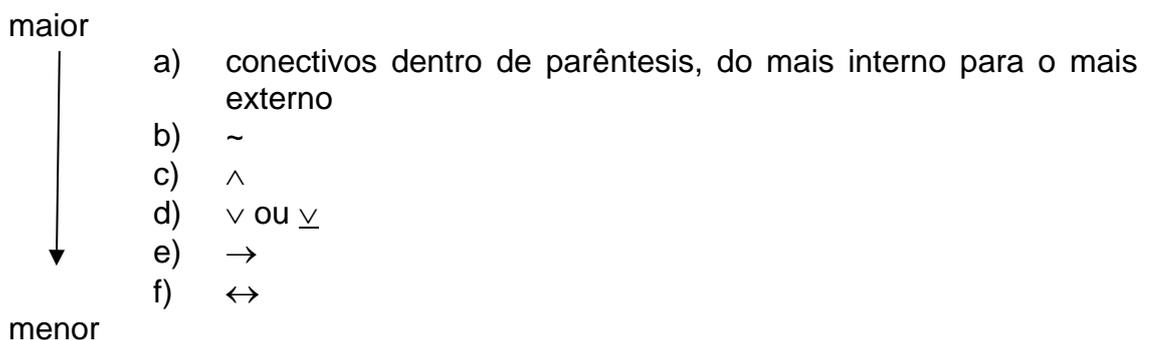
p	~p
V	F
F	V

2.4.5 – Ordem De Precedência Dos Conectores

Para reduzir o número de parêntesis necessários em uma proposição composta (fórmula lógica proposicional), estipula-se uma ordem na qual os conectores são aplicados. A ordem de precedência é:

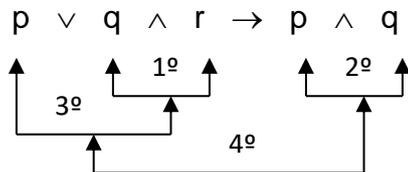
maior

a) conectivos dentro de parêntesis, do mais interno para o mais externo
b) ~
c) $\wedge$
d) $\vee$ ou $\veebar$
e) $\rightarrow$
f) $\leftrightarrow$

menor

Quando há dois ou mais conectivos de mesma ordem de precedência, o conectivo mais a esquerda na fórmula proposicional tem prioridade sobre o conectivo à direta.

Exemplos:
a) "p $\vee$ ~ q " equivale a " (p $\vee$ (~ q)) "

b) "$p \vee q \rightarrow r$" equivale a "$((p \vee q) \rightarrow r)$"

c) "$p \vee q \wedge r \rightarrow p \wedge q$" equivale a "$((p \vee (q \wedge r)) \rightarrow (p \wedge q))$"

2.5 - Exercícios

2.5.1 - Represente as seguintes frases em linguagem natural, utilizando lógica formal. Para cada resposta, devem-se especificar as proposições simples extraídas:

a) João mede 1,78m e Maria pesa 60kg.

b) Fortaleza é capital do Maranhão desde que Rio de Janeiro tenha mais de 250 mil habitantes.

c) Ontem o dólar não fechou a R$2,18 ou o índice Bovespa fechou estável.

d) Só irei ao clube se amanhã fizer sol.

2.5.2 - Sejam as seguintes proposições simples:

p: "Tiradentes morreu afogado" e q: "Jaime é gaúcho".

Traduzir para linguagem natural, as seguintes proposições compostas:

a) $p \wedge q$

b) $\sim p \vee q$

c) $q \vee \sim p$

2.5.3 - Determinar o valor lógico (V ou F) de cada uma das seguintes proposições.

a) O número 23 é primo.

b) Goiânia é a capital de Tocantins.

c) O número 25 é quadrado perfeito.

d) Todo número divisível por 5 termina com 5.

e) $\pi = 3$

f) $3 > 2$

g) $\pi > 3$

h) $\sqrt{4} = 2$

2.5.4 - Sejam as proposições:

p: João joga futebol.

q: Pedro joga tênis.

Traduzir as formulas lógicas para o português.

a) $p \vee q$ b) $p \wedge q$ c) $p \wedge \sim q$
d) $\sim p \wedge \sim q$ e) $\sim\sim p$ f) $\sim(\sim p \wedge \sim q)$

2.5.5 - Determinar o valor lógico (V ou F) de cada uma das proposições compostas abaixo, sabendo o valor lógico de cada proposição simples v(p) = V e v(q) = F.

a) $p \vee q \vee p$ b) $p \wedge q \vee p$ c) $p \wedge \sim q$ d) $p \vee (\sim p \wedge \sim q)$
e) $\sim p \wedge \sim q$ f) $\sim\sim p$ g) $\sim p \wedge q \vee p$ h) $p \wedge (\sim q \wedge p)$

2.2.6 - Construa a tabela-verdade de cada uma das seguintes proposições:

a) $p \vee q \vee p$ b) $p \wedge q \vee p$ c) $p \wedge \sim q$
d) $\sim p \wedge \sim q$ e) $\sim\sim p$ f) $p \vee (\sim p \wedge \sim q)$
g) $(p \wedge \sim q) \vee (r \vee s) \wedge p \vee r$

3 - MÉTODOS PARA DETERMINAÇÃO DA VALIDADE DE FÓRMULAS

São métodos usados para determinar ou verificar se a fórmula lógica proposicional é válida ou quais os valores lógicos apresentados. Existem vários métodos.

3.1 - Método da Tabela-Verdade

O método da Tabela-verdade é o método da força bruta utilizado na determinação da validade de fórmulas da lógica proposicional. No método da tabela-verdade são consideradas todas as possibilidades de valores de verdade associados a esses símbolos proposicionais. Na coluna de resultado, observamos uma das três soluções: Tautologia, Contradição ou Contingência.

Tautologia - denomina-se tautologia a proposição composta que é sempre verdadeira. Na tabela-verdade de uma proposição tautológica, a última coluna (à direita) contém somente V’s (verdade).

Contradição - denomina-se contradição a proposição composta que é sempre falsa. Na tabela-verdade de uma proposição contraditória, a última coluna (à direita) contém somente F’s (falsidade).

Contingência - denomina-se contingência a proposição composta que pode ser verdadeira e pode ser falsa. Na tabela-verdade de uma proposição contigencial, a última coluna (à direita) contém V’s (verdadeiros) e F’s (falsidades), cada um pelo menos uma vez.

Exemplos:

1) Demonstração de uma tautologia: $p \vee \sim(p \wedge q)$

p	q	$p \wedge q$	$\sim (p \wedge q)$	$p \vee \sim (p \wedge q)$
V	V	V	F	V
V	F	F	V	V
F	V	F	V	V
F	F	F	V	V

2) Demonstração de uma contingência: $(p \rightarrow q) \leftrightarrow (\sim p \rightarrow \sim q)$

p	q	$p \rightarrow q$	$\sim p$	$\sim q$	$\sim p \rightarrow \sim q$	$(p \rightarrow q) \leftrightarrow (\sim p \rightarrow \sim q)$
V	V	V	F	F	V	V
V	F	F	F	V	V	F
F	V	V	V	F	F	F
F	F	V	V	V	V	V

Como podemos ver, a fórmula é uma contingência.

3.2 - Exercício:

Determinar quais das proposições abaixo são tautologia, contradição ou contingência.

a) $(p \vee (p \wedge q)) \leftrightarrow p$
b) $(\sim p \wedge \sim q) \vee (p \rightarrow q)$
c) $(p \vee \sim q) \leftrightarrow ((p \rightarrow r) \vee q)$
d) $(\sim p \vee \sim q) \leftrightarrow (p \wedge q)$
e) $p \vee (\sim p \leftrightarrow q)$
f) $(\sim p \vee q) \leftrightarrow (p \rightarrow q)$

4 – CONSTRUINDO UM ALGORITMO

Segundo Pereira (2009), existem diversas formas de escrever um algoritmo, podendo ser citadas o pseudocódigo (ou português estruturado), fluxograma, diagrama de Chapin e descrição narrativa.

A descrição narrativa não é muito utilizada em informática porque pode ser ambígua e dar margem a interpretações erradas (PEREIRA, 2009).

Os dois tipos mais comuns são o pseudocódigo que utiliza uma forma mais estruturada, assemelhando-se àquelas utilizadas pelas linguagens de programação e o fluxograma que emprega figuras geométricas para ilustrar os passos a serem seguidos (PEREIRA, 2009).

O Diagrama de Chapin, Diagrama Nassi-Shneiderman ou Diagrama N-S mostra a solução por meio de quadros organizados hierárquica e estruturada. Este tipo não é muito utilizado, pois vários procedimentos tornam-se difíceis de serem mostrados por meio deste diagrama (PEREIRA, 2009).

Exemplo. Dado o problema a seguir. A partir de 3 notas de um aluno, calcular sua média aritmética e escrever na tela se ele foi aprovado ou reprovado, levando em conta que a média para aprovação deve ser pelo menos 5.0 (PUCRJ, 2010).

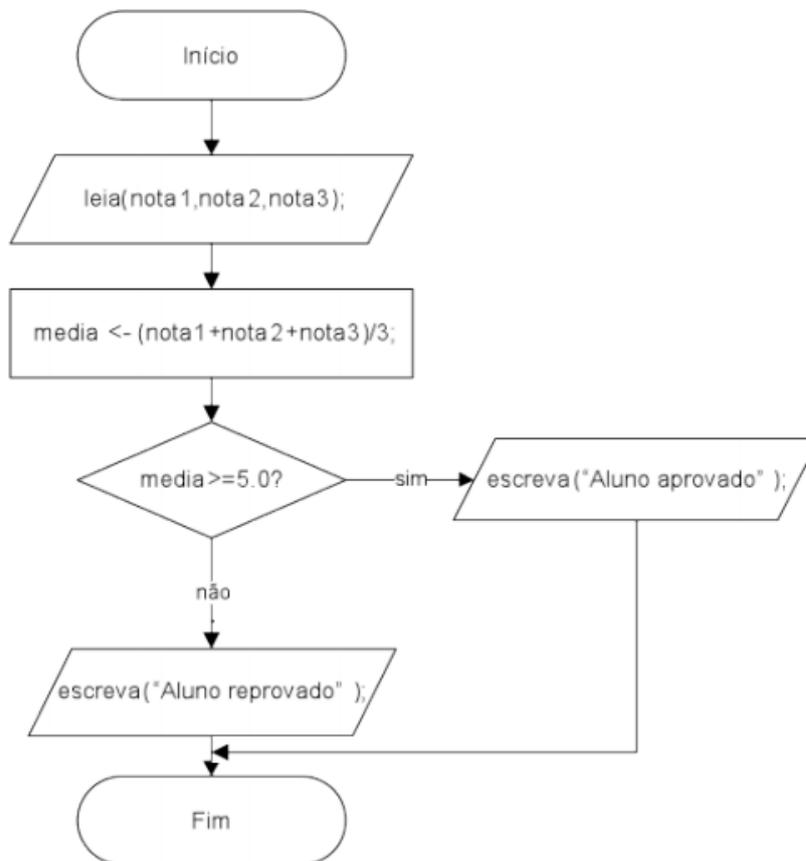
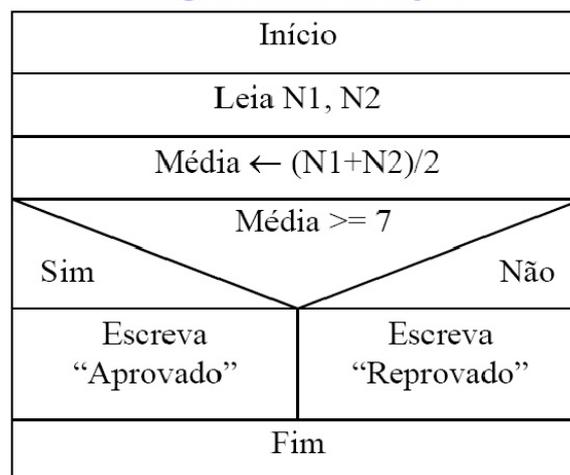
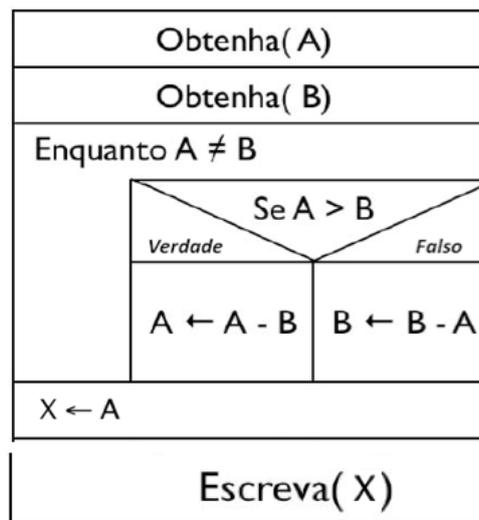
Soluções:

a) Descrição narrativa ou Linguagem natural (PUCRJ, 2010):

1. Obter as 3 notas das provas do aluno
2. Calcular a média aritmética das 3 notas
3. Comparar a média com o valor 5.0
4. Se for maior ou igual, escrever “aprovado”
5. Caso contrário, escrever “reprovado”

b) Pseudocódigo (PUCRJ, 2010):

```
Algoritmo Exemplo
   variaveis
     real: media, nota1, nota2, nota3
   inicio
     leia ( nota1, nota2 e nota3)
     media = (nota1+nota2+nota3)/3
     se (media >= 5) entao
        escreva (“aluno aprovado”)
     senao
        escreva (“aluno reprovado”)
     fimse
   fim
```

c) Fluxograma (PUCRJ, 2010):

d) Diagrama de Chapin (MAGALHÃES, 2007):

e) Testando a solução

Considerando o algoritmo apresentado, é possível testar se o mesmo atende a solicitação. Neste caso, são lidas 3 notas, calculada a média e impresso o resultado (media maior ou igual a 5 é aprovado e caso contrário é reprovado).

```
Algoritmo Exemplo
   variaveis
     real: media, nota1, nota2, nota3
   inicio
     leia ( nota1, nota2 e nota3)
     media = (nota1+nota2+nota3)/3
     se (media >= 5) entao
       escreva (“aluno aprovado”)
     senao
       escreva (“aluno reprovado”)
     fimse
   fim
```

Passo	Nota1	Nota2	Nota3	Media	Impressão
1 (leia)	7	5	6	-	-
1	7	5	6	6	-
1 (se)	7	5	6	6	Aluno aprovado
1 (leia)	7	3	3	-	-
1	7	3	3	4,3	-
1 (se)	7	3	3	4,3	Aluno reprovado

Assim, foi testado para todas as situações possíveis e a proposta de solução atende o exercício.

4.1 – Fluxograma

Segundo Emiliano (2015), os símbolos utilizados para a construção de um fluxograma são:

Símbolo	Função
TERMINAL	Indica o INÍCIO ou FIM de um processamento Exemplo: Início do algoritmo
PROCESSAMENTO	Processamento em geral Exemplo: Calculo de dois números
ENTRA/SAÍDA	Operação de entrada e saída de dados Exemplo: Leitura e Gravação de Arquivos
DECISÃO	Indica uma decisão a ser tomada Exemplo: Verificação de Sexo
DESVIO	Permite o desvio para um ponto qualquer do programa
ENTRADA MANUAL	Indica entrada de dados através do Teclado Exemplo: Digite a nota da prova 1
EXIBIR	Mostra informações ou resultados Exemplo: Mostre o resultado do calculo
RELATÓRIO	Relatórios

Fonte: Emiliano (2015)

4.2 – Pseudocódigo

O formato básico do pseudocódigo em Portugol (adaptado de FORBELONE; EBERSPACHER, 2005) pode ser apresentado como:

a) sintaxe básica de pseudocódigo:

```
algoritmo <nome>
  {Área de declaração de variáveis e constantes}
  {Área de procedimentos e funções}
  inicio
     {Área de comandos}
  fim
```

b) Operadores Aritméticos

pot (potenciação), rad (radiciação)
*, /, div (divisão inteira), mod (resto da divisão inteira)
+, -
A precedência segue a ordem em que foram apresentados acima.

Exemplo, dada a expressão ***x - mod(x, y) * 2 + pot(z, 2)*** qual o resultado para os valores x = 7, y = 2 e z = 3? Resolvendo passo a passo, temos:

x - mod(x, y) * 2 + pot(z, 2)

7 – mod(7, 2) * 2 + pot(3, 2)

7 – mod(7, 2) * 2 + 9

7 – 1 * 2 + 9

7 – 2 + 9

14

c) Operadores Relacionais

= (igual)	3 = 3, x = y
< (menor que)	5 < 6, x < y
> (maior que)	5 > 4, x > y
<= (menor ou igual a)	5 <= 6, x <= y
>= (maior ou igual a)	5 >= 4, x >= y
<> (diferente de)	8 <> 9, x <> y

Exemplo, dada a expressão ***mod(5,2) + 3 <> div(15,4)*** qual o valor lógico resultante? Resolvendo passo a passo, temos:

mod(5, 2) + 3 <> div(15, 4)

1 + 3 <> 3

4 <> 3

V

d) Operadores Lógicos

ou, nao, e, xou

e) Entrada de Dados

leia

f) Saída de Dados

escreva

g) Estruturas de repetição

```
enquanto <expressão-lógica> faca
   <sequência-de-comandos>
fimenquanto

para <variável> de <valor-inicial> ate <valor-limite> [passo <incremento>] faca
   <sequência-de-comandos>
fimpara

repita
   <sequência-de-comandos>
ate <expressão-lógica>
```

h) Estruturas de seleção ou Condicional

```
se <expressão-lógica>
  entao
    <sequência-de-comandos-1>
  senao
    <sequência-de-comandos-2>
fimse

se <expressão-lógica>
  entao
    <sequência-de-comandos-1>
fimse

escolha <expressão-de-seleção>
   caso <situacao1>:
     <sequência-de-comandos-1>
   caso <situacao2>:
      <sequência-de-comandos-2>
   caso <situacaon>:
      <sequência-de-comandos-n>
  caso contrario
     <sequência-de-comandos-extra>
fimescolha
```

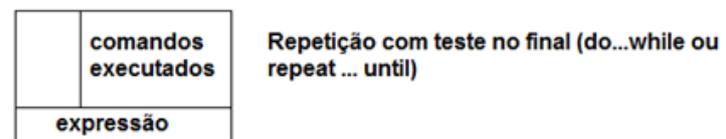
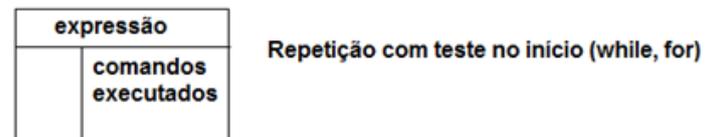
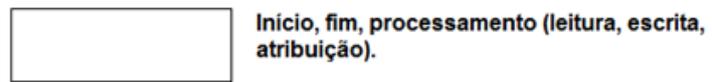
i) Variáveis e constantes

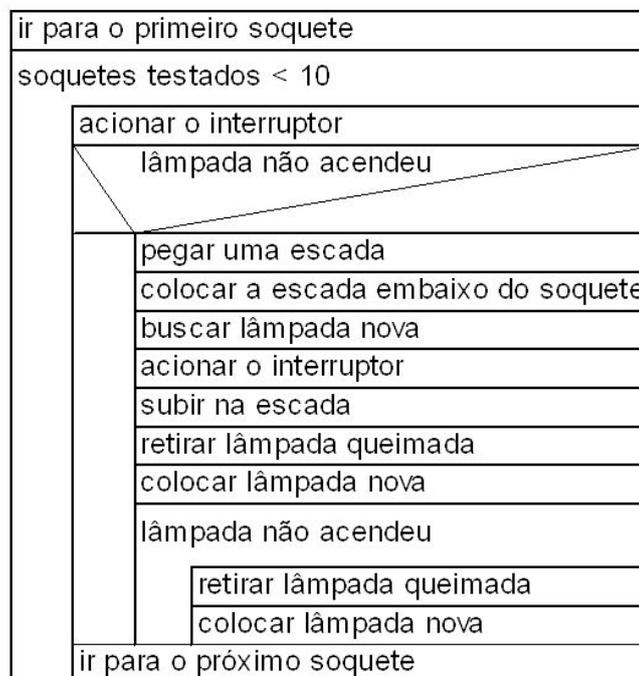
Variáveis quando um dado tem a possibilidade de ser alterado em algum instante do decorrer do tempo de execução do algoritmo. As variáveis podem ser do tipo inteiro, real, lógico e caracter.

Constantes são dados que não sofre nenhuma variação no decorrer do tempo de execução do algoritmo. Podem ser do tipo inteiro, real, lógico e caracter.

4.3 – Diagrama de Chapin

Segundo Texera, Silva e Muniz (2013) o diagrama permite a visualização do fluxo lógico do algoritmo e é voltado para a programação estruturada. A simbologia utilizada no Diagrama de Chapin é descrita como:

Símbolo	Descrição
(retângulo)	**Início, fim, processamento (leitura, escrita, atribuição).**
expressão / V / F	**Decisão (if)**
expressão / comandos executados	**Repetição com teste no início (while, for)**
comandos executados / expressão	**Repetição com teste no final (do...while ou repeat ... until)**

É a substituição do fluxograma tradicional por diagrama que apresenta uma visão hierárquica e estruturada da lógica do programa. Sua maior vantagem é a representação das estruturas que tem um ponto de entrada e um ponto de saída e são compostas pelas estruturas básicas de controle de sequência, seleção e repartição (TEXERA; SILVA; MUNIZ, 2013).

Exemplo:

Fonte: Montebello Júnior (2013)

4.4 – Exercícios

a) Supondo que há 4 variáveis do tipo inteiro (A, B, C, D) cujos valores são 2, 5, 6 e 10 respectivamente. Qual o resultado das operações aritméticas abaixo?

i) A + B – C

ii) 2 * A + 3 * B – C

iii) B * A - C / 3 + D

iv) B * (A - C / 3) + D

v) D - mod(B, A) * 2 + C

vi) pot(C, A) / 4 + (C – 3)

b) Supondo que há 4 variáveis do tipo inteiro (A, B, C, D) cujos valores são 5, 10, 15 e 20 respectivamente. Qual o resultado das operações logicas abaixo?

i) B = 5 * 2

ii) C > B + A

iii) B + C * 2 >= pot(A, 2) + C * 2

iv) B * mod(D, 3) <= C + A * 3

v) D - mod(B, A) * 2 + C <> pot(A, 3) – 3 * C

c) Construir o algoritmo que calcula o custo final de um produto que é composto de 3 matérias primas. Inicialmente são lidas os valores dos três materiais e a fórmula é:

Custo = M01 + 3 * M02 + 2 * M03, onde M0x é uma matéria prima.

Imprimir o valor do custo final. Resolver utilizando fluxograma e pseudocódigo.

d) Escrever um algoritmo que lê dois valores inteiros e imprime o resultado para as quatro operações básicas (soma, subtração, divisão e multiplicação). Resolver utilizando fluxograma e pseudocódigo.

e) Escrever um algoritmo para calcular e imprimir o volume de uma esfera de raio R que é fornecido pelo usuário. O volume de uma esfera é dado pela formula: $v=4(\pi R^3)/3$. Resolver utilizando fluxograma e pseudocódigo.

f) Escrever um algoritmo que lê um número inteiro e escreve se ele é par ou impar. Resolver utilizando fluxograma e pseudocódigo.

g) Escrever um algoritmo que lê o peso e a altura de uma pessoa adulta, calcula e imprime o Índice de Massa Corporal, cuja fórmula é [IMC = peso / ($altura^2$)]. Resolver utilizando fluxograma e pseudocódigo.

h) Escrever um programa que imprime a área de um retângulo, tendo lido os valores da base e da altura. Resolver utilizando fluxograma e pseudocódigo.

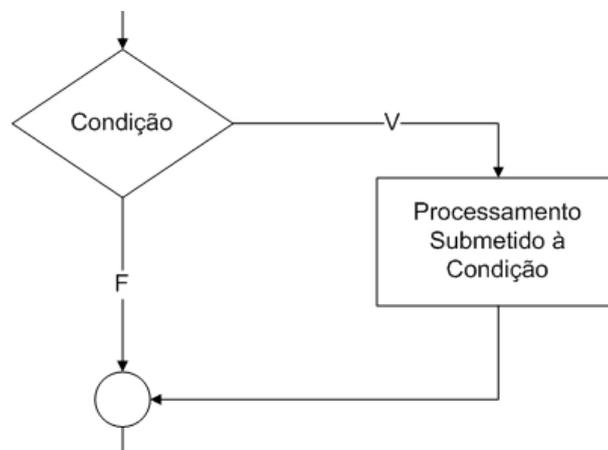
i) Escrever um programa que imprime a área de um triangulo, tendo lido os valores da base e da altura. Resolver utilizando fluxograma e pseudocódigo.

5 - ESTRUTURAS CONDICIONAIS (Ou SELEÇÃO)

A estrutura condicional possibilita a escolha de uma ação a ser executada quando determinadas condições são ou não satisfeitas. Ela quebra a estrutura sequencial, pois condiciona a execução da instrução ou bloco a uma dada condição.

A Estrutura Condicional pode ser Simples ou Composta.

5.1 – Estrutura Condicional Simples

A estrutura condicional simples expõe que a instrução ou bloco de instruções só serão executados se a condição for verdadeira.

Fonte: Wo; Costa; Santos (2009)

Notação:

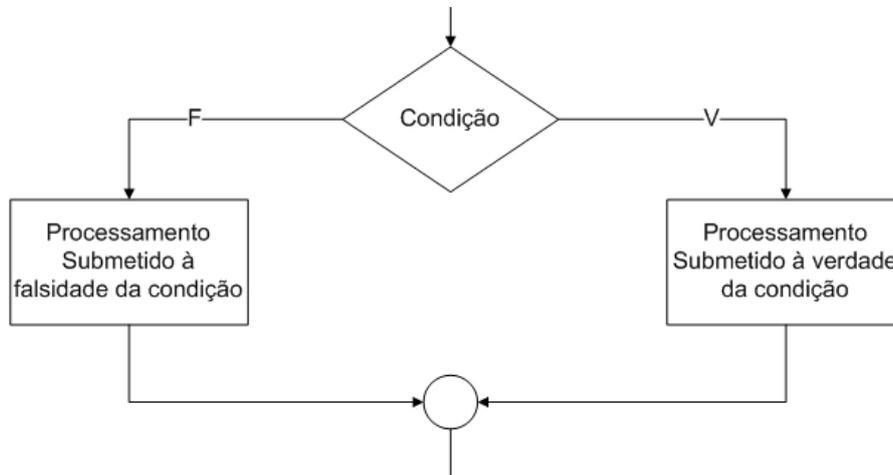
```
se <expressão-lógica>
  entao
    <sequência-de-comandos-1>
fimse
```

Exemplo:

```
se idade >= 18
  entao
    escreva ("É maior de idade")
fimse
```

5.2 – Estrutura Condicional Composta

A estrutura condicional composta define que uma instrução ou bloco de instruções só será executada se uma condição for satisfeita e outra instrução ou bloco só será executada se a condição não for satisfeita, ou seja, diferentemente da

condicional simples, a estrutura condicional composta executa um bloco de instruções se a condição for verdadeira e outro se a condição for falsa.

Fonte: Wo; Costa; Santos (2009)

Notação:

```
se <expressão-lógica>
  entao
    <sequência-de-comandos-1>
  senao
    <sequência-de-comandos-2>
fimse
```

Exemplo:

```
se idade >= 18
  entao
    escreva ("É maior de idade")
  senao
    escreva ("É menor de idade")
fimse
```

5.3 - Encadeamento de estruturas condicionais

Um recurso muito utilizado em programação é o encadeamento de estruturas condicionais. A estrutura sequencial se encontrar dentro das outras estruturas. Quanto ao encadeamento de estruturas condicionais, pode-se tanto uma estrutura simples encadear uma composta quanto uma composta pode encadear uma ou mais estruturas simples.

Exemplo:

```
se idade >= 18
  então
      se sexo = "M"
```

```
            então
                escreva (“Voce já alistou?”)
            fimse
      senao
        escreva (“É menor de idade”)
    fimse
```

5.4 - Seleção Múltipla

Permite escolher uma entre várias alternativas expressas por valor inteiro ou caractere.

Na seleção múltipla, a expressão é calculada e os comandos relacionados abaixo da constante com o mesmo valor da expressão são executados. Se não houver valor igual ao da expressão, os comandos subordinados à palavra senão são executados. A cláusula senão é opcional. No máximo uma das opções é executada.

Notação:

```
escolha <expressão-de-seleção>
   caso <situacao1>:
     <sequência-de-comandos-1>
   caso <situacao2>:
     <sequência-de-comandos-2>
   caso <situacaon>:
     <sequência-de-comandos-n>
  caso contrario
     <sequência-de-comandos-extra>
fimescolha
```

Exemplo:

```
escreva("Entre 1º número: ")
leia(nro1)
escreva("Entre 2º número: ")
leia(nro2)
escreva("Opção: ")
leia(op)
escolha op
     Caso 1:
         escreva("Soma = ",nro1 + nro2)
     Caso 2:
         escreva("Subtração = ",nro1 - nro2)
     Caso 3
         escreva("Multiplicação = ",nro1 * nro2)
     Caso 4
         escreva("Divisão = ",nro1 / nro2)
```

```
        Caso 5
            escreva("Fim")
            interrompa
        caso contrario
            escreva("opção invalida")
    fimescolha

    Escreva (“digite o time”)
    Leia (time)
    Escolha (time)
        Caso “Botafogo”,”Flamengo”,“Fluminense”,”Vasco”
            Escreva(“Time carioca”)
        Caso “Corinthians”,”Palmeiras”,“Santos”,”São Paulo”
            Escreva(“Time paulista”)
        Caso “America”,“Atletico”,”Cruzeiro”,“Uberlândia”
            Escreva(“Time mineiro”)
        CasoContrario
            Escreva(“Time de outros estados”)
    FimEscolha
```

5.5 – Exercícios

Resolver os algoritmos utilizando fluxograma e pseudocódigo.

1) Escrever um algoritmo que lê dois números inteiros e imprime qual é o maior deles.

2) Escrever um algoritmo que recebe duas informações através do teclado, a altura (h, em metros) e o sexo da pessoa. E com estas informações calcule o peso ideal para esta pessoa, utilizando as seguintes fórmulas:

   Para homens, Peso = (72,7 * h) – 58;

   Para mulheres, Peso = (62,1 * h) – 44,7

3) Escrever um algoritmo que lê três números inteiros e imprime qual é o menor deles e também a média destes números.

4) Escrever um algoritmo que lê três números inteiros e imprime qual é o maior deles e também a soma dos outros dois número.

5) Escrever um algoritmo que calcula o Índice de Massa Corporal [IMC = peso / (altura^2)] para uma pessoa adulta e a partir desta informação, indique qual a sua situação, onde:

   IMC < 18,5 → abaixo do peso

   18,5 <= IMC <= 25,0 → peso normal

   25,0 < IMC <= 30,0 → acima do peso (sobrepeso)

   IMC > 30,0 → obeso

6) Fornecido o consumo de energia elétrica em Kwh, escreva um algoritmo para calcular a conta de energia elétrica de um estabelecimento, considerando a seguinte tabela de valores:

Consumo em Kwh	Preço do Kwh em R$
Até 300	1,25
De 301 até 600	1,50
De 601 até 1000	1,75
Acima de 1000	2,00

7) Fornecido o sexo, a altura (em metros) e peso (em Kg) de uma pessoa, escrever um algoritmo que calcula o peso ideal seguindo a seguinte fórmula:

Para o sexo feminino, PesoIdeal = (62,1 * altura) – 48,7

Para o sexo masculino, PesoIdeal = (72,7 * altura) – 62,0

Comparar o peso da pessoa com o peso ideal e escrever as seguintes mensagens:

Se a diferença do peso for maior que 6 kg do peso ideal, escrever "Alerta de diferença de peso maior que 6 Kg";

Se a diferença do peso for menor que 6 kg do peso ideal, escrever "Esta dentro da margem de peso"

Se o peso for igual ao peso ideal, escrever "Peso ideal".

8) Considerando a tabela de produtos abaixo, construir um algoritmo que lê o código e a quantidade de produtos e calcula o valor da venda. E em seguida imprime a descrição do produto e o valor total calculado.

Código	Descrição	Preço (R$)
1	Caneta	1,20
2	Lapis	0,80
3	Borracha	1,00
4	Régua	1,50

9) Fornecido a distância percorrida em Km por um carro alugado, escreva um algoritmo para calcular o custo mensal da locação, considerando que há três valores a serem considerados. O custo do combustível (ler o valor), considerando que o veículo faz 14 Km/litro. A manutenção e a locação são calculadas por distância percorrida (em Km) conforme a tabela:

Distância (Km)	Manutenção em R$ (por Km percorrido)	Locação em R$ (por Km percorrido)
Até 600	1,00	5,00
De 601 até 1200	0,95	4,70
De 1201 até 2000	0,88	4,20
Acima de 2000	0,80	4,00

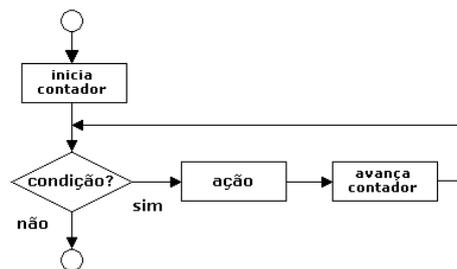
6 - ESTRUTURAS DE REPETIÇÃO

Permitem executar mais de uma vez um mesmo trecho de código. Trata-se de uma forma de executar blocos de comandos somente sob determinadas condições, mas com a opção de repetir o mesmo bloco quantas vezes forem necessárias.

As estruturas de repetição são úteis, por exemplo, para repetir uma série de operações semelhantes que são executadas para todos os elementos de uma lista ou de uma tabela de dados, ou simplesmente para repetir um mesmo processamento até que uma certa condição seja satisfeita.

Há três modelos de estrutura de repetição: repetição com variável de controle, repetição com teste no início e repetição com teste no final.

6.1 Repetição com variável de controle

É uma estrutura de controle de fluxo de execução que sempre repete uma quantidade determinada de vezes um mesmo trecho do algoritmo.

Fonte: LIMA (2019)

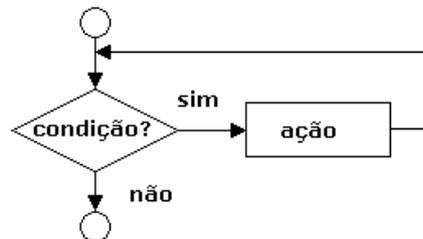
Notação:

```
Para <variável-controle> de <valor-inicial> ate <valor-final> faca
     <sequência-de-comandos>
Fimpara
```

Exemplo:

```
soma <- 0
para j de 1 ate 7 faca
   escreva("Entre com o número: ")
   leia(nro)
   soma <- soma + nro
fimpara
escreva("Soma: ",soma)
escreva("Media: ",soma/7)
```

6.2 Repetição com teste no início

É uma estrutura de controle de fluxo de execução que permite repetir diversas vezes um mesmo trecho do algoritmo, porém verificando antes de cada execução se deve ser executado o trecho.

Fonte: LIMA (2019)

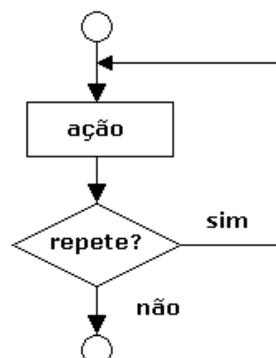
Notação:

```
Enquanto <condição> faça
    <sequência-de-comandos>
Fimenquanto
```

Exemplo:

```
j <- 1
enquanto j <= 10 faca
      escreva (j:3)
      j <- j + 1
fimenquanto
escreval("fim")
```

6.3 Repetição com teste no final

É uma estrutura de controle de fluxo de execução que permite repetir diversas vezes um mesmo trecho do algoritmo, verificando ao final de cada execução se deve ser novamente executado o trecho.

Fonte: LIMA (2019)

Notação:

```
Repita
    <sequência-de-comandos>
Ate <expressão-logica>
```

Exemplo:

```
j <- 1
repita
    escreva (j:3)
    j <- j + 1
ate j > 10
escreva("fim")
```

6.4 – Exemplo de Menu

Utilizando a estrutura de controle **repita** para organizar um menu.

```
algoritmo "Calculadora com menu"
// Função: programa que simula uma calculadora com menu
// Autor : Walteno Martins Parreira Jr
// Data : 20/04/2017
// Seção de Declarações
var
  op,v1,v2: inteiro
  nro1, nro2: real
inicio
// Seção de Comandos
repita
  // apresenta o menu da calculadora
  escreval(" -- Calculadora --")
  escreval("1 - soma")
  escreval("2 - subtração")
  escreval("3 - multiplicação")
  escreval("4 - divisão")
  escreval("9 - sair")
  escreva("Opção: ")
  leia(op)
  // faz a seleção da opção escolhida
  escolha op
     Caso 1
         // opção soma
         escreva("Entre 1º número: ")
         leia(nro1)
         escreva("Entre 2º número: ")
         leia(nro2)
         escreval("Soma = ",nro1 + nro2)
     Caso 2
         // opção subtração
```

```
            escreva("Entre 1º número: ")
            leia(nro1)
            escreva("Entre 2º número: ")
            leia(nro2)
            escreval("Subtração = ",nro1 - nro2)
        Caso 3
            // opção multiplicação
            escreva("Entre 1º número: ")
            leia(nro1)
            escreva("Entre 2º número: ")
            leia(nro2)
            escreval("Multiplicação = ",nro1 * nro2)
        Caso 4
            // opção divisão
            escreva("Entre 1º número: ")
            leia(nro1)
            escreva("Entre 2º número: ")
            leia(nro2)
            escreval("Divisão = ",nro1 / nro2)
        Caso 5 ate 10
            // opção sair
            escreval("Fim")
            interrompa
        outrocaso
            // opção indicada inexistente
            escreval("opção invalida")
      fimescolha
      // volta para o inicio e mostra o menu
      // como não tem um criterio de parada, fica executando indefinidamente
fimrepita
fimalgoritmo
```

6.5 – Exercícios

Resolver os algoritmos utilizando fluxograma e pseudocódigo.

Usar a estrutura de repetição **Para**

1) Escrever um algoritmo que lê um número inteiro e imprime os próximos dez números.

2) Escrever um algoritmo que lê um número inteiro e imprime o valor resultante deste número elevado a quinta potência, mas sem utilizar a função potenciação.

3) Escrever um algoritmo que lê dois números inteiros e imprime o valor resultante do primeiro número lido elevado potência do segundo número lido, mas sem utilizar a função potenciação.

3) Escrever um algoritmo que lê dois números inteiros, sendo o segundo maior que o primeiro, e imprime todos os números existentes entre eles.

4) Escrever um algoritmo que lê cinco números inteiros, um de cada vez, e imprime qual é o maior e o menor dos números lidos e também a soma dos números lidos.

5) Escrever um programa que imprime um elemento da sequência de Fibonacci, dado o número do elemento.

6) Escrever um algoritmo que lê um número inteiro e imprime todos os números pares existentes de 1 até o número lido. Considere que tem uma função denominada ***ePar(nro)*** que devolve Verdade se ele for par ou use a função ***mod(nro,2)*** que devolve zero se for par.

7) Escrever um algoritmo que lê dois números inteiros e imprime todos os números primos existentes entre eles. Considere que tem uma função denominada ***ePrimo(nro)*** que devolve Verdade se ele for primo.

8) Escrever um algoritmo que lê um número inteiro e imprime os próximos dez números pares existentes.

Usar a estrutura de repetição **Enquanto**

9) Escrever um algoritmo que lê uma sequência de números inteiros até que seja lido o valor zero e imprime a soma dos números lidos.

10) Escrever um algoritmo que lê uma sequência de números inteiros até que seja lido o valor zero e imprime a média dos números pares lidos.

11) Escrever um algoritmo que lê uma sequência de números inteiros até que seja lido o valor zero e imprime a média dos números lidos e o menor número lido diferente de zero que foi critério de parada.

12) Escrever um algoritmo que lê a quantidade não informada de pessoas que estão sendo monitoradas no posto de saúde. Para cada pessoa é anotada o peso e a altura (em metros), que é fornecida via teclado, sendo que o peso igual a zero informa o final da leitura. Calcule e escreva: a) A maior altura informada, b) A menor altura informada, c) A média dos pesos informados.

13) Escrever um algoritmo que lê a quantidade não informada de dados de pessoas que estão sendo monitoradas no posto de saúde. Para cada pessoa é anotada o sexo (M ou F) e a altura (em metros), que é fornecida via teclado, sendo que a altura igual a zero informa o final da leitura. Calcule e escreva: a) A maior altura informada, b) A media das alturas das mulheres, c) A quantidade de homens, d) A menor altura dos homens.

7 - ARRANJOS UNIDIMENSIONAIS E BIDIMENSIONAIS

Arranjos são estrutura de dados que permitem o armazenamento de um conjunto de dados do mesmo tipo. É uma coleção de elementos do mesmo tipo no qual a posição de cada elemento está definida de forma única por um valor do tipo inteiro. Podem ser:

a) Unidimensionais - contém apenas um índice;

b) Multidimensionais – contém dois ou mais índices.

Considerando que a memória é unidimensional e pode ser considerada como armazenando palavras numeradas de 1 até *m* (ou, de 0 a *m*-1).

Assim, o objetivo é a representação de arranjos *n* dimensionais em uma memória unidimensional, logo tem-se que:

n=1: vetor (arranjo unidimensional)

n=2: matriz (arranjo bidimensional)

n>2: arranjo (ou matriz) multidimensional

Conceitualmente, caracterizam-se por:

Conjunto de elementos de um mesmo tipo. Logo, todos os elementos de um vetor devem ser do mesmo tipo de dado: inteiro, caractere, etc ...;

Elementos são referenciados por um único nome e individualizados pela posição que ocupam no conjunto. Os vetores permitem que seja definido uma série de variáveis com o mesmo nome, diferenciando-as através de um número inteiro chamado índice;

Usa-se um número inteiro, chamado de índice para acessar cada elemento do vetor. Na maioria das linguagens, os índices iniciam em 0 e vão até n – 1, para índice de tamanho n

A quantidade de variáveis definidas pelo índice (número inteiro) determina o tamanho do vetor.

7.1 Vetores

Vetores podem ser definidos como posições de memória identificadas por um mesmo nome, individualizadas por índices e cujo conteúdo é do mesmo tipo. São unidimensionais, pois contém apenas um índice para individualizar uma posição de memória.

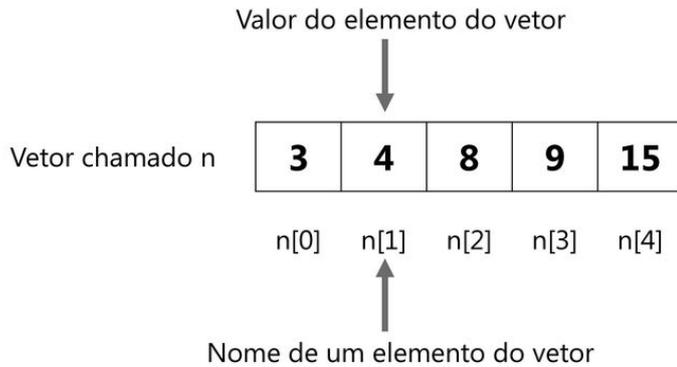
Simbolicamente:

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_n \end{bmatrix}$$

Fonte: Adaptado de Novaes (2019)

Notação:

<variável> : vetor [<valor-inicial>..<valor-final>] de <tipo-variavel>

Esquema:

Exemplo:

Programa que mostra a definição e manipulação de um vetor.

```
Algoritmo "Vetores"
Var
  // Seção de Declarações das variáveis
  v: vetor[1..10] de inteiro
  nv: vetor[1..10] de caracter
  i, n, soma: inteiro
  nx: caracter
Inicio
  // Seção de Comandos, procedimento, funções, operadores, etc...
  para i de 1 ate 5 faca
    escreva("Digite o ",i,"º nome desejado ")
    leia(nx)
    nv[i] <- nx
    escreva("Digite o ",i,"º valor desejado ")
    leia(n)
    v[i] <- n
    soma<-soma+n
  fimpara
  escreval(" A soma dos valores é: ",soma)
  para i de 1 ate 5 faca
    escreva("Nome do item[",i,"]: ", nv[i])
    escreval(" Valor do item[",i,"]: ", v[i])
  fimpara
Fimalgoritmo
```

7.2 Matrizes

Matrizes podem ser definidos como posições de memória identificadas por um mesmo nome, individualizadas por índices e cujo conteúdo é do mesmo tipo.

São multidimensionais, pois contém mais de um índice para individualizar uma posição de memória. Particularmente, será apresentada com duas dimensões.

Em Matemática, matrizes são organizações de informações numéricas em uma tabela retangular formada por linhas e colunas. Essa organização em uma tabela facilita que se possa efetuar vários cálculos simultâneos com as informações contidas na matriz. Toda matriz tem o formato m x n (leia-se: m por n, com n e m pertencente aos números naturais maiores que zero), onde m é o número de linhas e n o número de colunas (NOVAES, 2019).

Simbolicamente:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

Fonte: Novaes (2019)

Notação:

<variável> : vetor [<valor-1>, <valor-2>] de <tipo-variavel>

Exemplo:

Programa que mostra a definição e manipulação de uma matriz.

```
Algoritmo "Matriz"
Var
  // Seção de Declarações das variáveis
  v: vetor[1..5,1..3] de inteiro
  i,n,soma,multiplo:inteiro
Inicio
  // Seção de Comandos, procedimento, funções, operadores, etc...
  para i de 1 ate 5 faca
    escreva("Digite o ",i,"º valor desejado ")
    leia(n)
    v[i,1] <- n
    soma <- 10+n
    v[i,2] <- soma
    multiplo <- 2*n
```

```
      v[i,3] <- multiplo
   fimpara
   escreval(" --- Impressão ---- ")
   para i de 1 ate 5 faca
      escreva("  valor[",i,",1]: ", v[i,1])
      escreva("  valor[",i,",2]: ", v[i,2])
      escreval("  valor[",i,",3]: ", v[i,3])
   fimpara
Fimalgoritmo
```

7.3 – Exercícios

Resolver os algoritmos utilizando fluxograma e pseudocódigo em VisualG.

Usar as estruturas Vetor e/ou Matriz na resolução

1) Escrever um algoritmo que lê cinco números inteiros, um de cada vez, armazena os números lidos em um vetor e a partir da leitura do vetor imprime qual é o maior e o menor dos números lidos e também a soma dos números lidos.

2) Escrever um algoritmo que lê a quantidade não informada de pessoas que estão sendo monitoradas no posto de saúde. Para cada pessoa é anotada o peso e a altura (em metros), que é fornecida via teclado, sendo que o peso igual a zero informa o final da leitura. Armazenar as informações em uma matriz e posteriormente, calcule e escreva: a) A maior altura informada, b) A menor altura informada, c) A média dos pesos informados.

3) Escrever um algoritmo que lê uma quantidade não informada de dados de pessoas que estão sendo monitoradas no posto de saúde. Para cada pessoa é anotada o sexo (M ou F) e a altura (em metros), que é fornecida via teclado, sendo que a altura igual a zero informa o final da leitura. Armazenar as informações em vetores e posteriormente, calcule e escreva: a) A maior altura informada, b) A média das alturas das mulheres, c) A quantidade de homens, d) A menor altura dos homens.

4) Escrever um algoritmo que lê uma quantidade informada inicialmente de notas de alunos da disciplina de Lógica. Para cada aluno são lidas quatro (4) notas variando de zero (0) a dez (10). Para ser aprovado o aluno necessita de média igual ou superior a seis (6). Ler as notas, calcular a média e armazenar em uma matriz. Considere o índice da linha da matriz como o número do aluno na chamada. Após o cálculo da média, ler os dados e escrever: "O Aluno y foi Aprovado (ou Reprovado) com média x,xx". No final informar: a) quantos alunos foram aprovados, b) quantos alunos foram reprovados, c) qual a porcentagem de alunos aprovados.

REFERENCIAS

CARBONI, Irenice de Fátima. **Lógica de Programação**. São Paulo: Pioneira Thomson Learning, 2003.

EMILIANO, Edson. **Diagrama de bloco**. Abr. 2015. Disponível em <https://www.edsonemiliano.com.br/blog/diagrama-de-bloco/>, Acesso em 17 fev. 2019.

FORBELONE, Andre Luiz Villar; EBERSPACHER, Henri Frederico. **Lógica de Programação**. 3. ed. São Paulo: Prentice Hall, 2005.

MAGALHÃES, Regis Pires. **Introdução à Lógica**. Abr. 2007. Disponível em <https://pt.slideshare.net/regispires/logica-algoritmo-02-algoritmo-presentation>, Acesso em 16 fev. 2019.

MOISES, Abreu. **Qual a importância de algoritmos na programação?** Nov. 2016. Disponível em <https://www.acadtec.com.br/blog/desenvolvimento-backend/qual-a-importancia-de-algoritmos-na-programacao>, Acesso em 16 fev. 2019.

MONTEBELLO JÚNIOR, Marco Antonio. Fundamentos de Lógica. 2013. Disponível em <https://slideplayer.com.br/slide/67235/>, Acesso em 20 fev. 2019.

MOREIRA, Rodrigo. **A importância do uso de Algoritmos no desenvolvimento intelectual e profissional**. Jul. 2017. Disponível em <https://www.profissionaisti.com.br/2017/07/a-importancia-do-uso-de-algoritmos-no-desenvolvimento-intelectual-e-profissional/>, Acesso em 16 fev. 2019.

NOVAES, Jean Carlos. **Matrizes:** Definições e Operações. 2019. Disponível em < https://matematicabasica.net/matrizes/>, acesso em 20 mai. 2019.

LIMA, Niels Fontes. **Ensino:** Física Geral e Experimental. 2019. Disponível em <http://www.ifba.edu.br/fisica/nfl/profNFL.html>, acesso em 18 fev. 2019.

PACIEVITCH, Ivan. **Lógica de Programação**. Info Escola. Disponível em <https://www.infoescola.com/informatica/logica-de-programacao/>, Acesso em 15 fev. 2019.

PEREIRA, Ana Paula. **O que é algoritmo?** TecMundo. Mai. 2009. Disponível em <https://www.tecmundo.com.br/programacao/2082-o-que-e-algoritmo-.htm>, Acesso em 15 fev. 2019.

PORFíRIO, Francisco. **O que é lógica?;** Brasil Escola. Disponível em <https://brasilescola.uol.com.br/filosofia/o-que-logica.htm>. Acesso em 15 fev. 2019.

PUCRJ – Pontifícia Universidade Católica do Rio de Janeiro. **Algoritmos e Pseudocódigo**. 2010. Disponível em <http://www.inf.puc-rio.br/~inf1005/material/slides/backup/2010_2/tema02_algoritmos.pdf>, Acesso em 16 fev. 2019.

TEXERA, Gabriel Giaretta; SILVA, Paula Francine; MUNIZ, Jonathan. **Fluxogramas, diagramas de blocos e de Chapin no desenvolvimento de algoritmos**. Devmedia, 2013. Disponível em <https://www.devmedia.com.br/fluxogramas-diagrama-de-blocos-e-de-chapin-no-desenvolvimento-de-algoritmos/28550>, Acesso em 20 fev. 2019.

WO, Celso; COSTA, Gustavo; SANTOS, Charles. **Algoritmos e Estruturas de Dados**. Centro Paula Souza. 2009. Disponível em <https://quetalumprograma.wordpress.com/estrutura-condicional/simples/>, acesso em 05 mar. 2019.

Licenciatura em Computação

Prof. Walteno Martins Parreira Júnior

www.waltenomartins.com.br

waltenomartins@iftm.edu.br